METHOD AND APPARATUS FOR FLEXIBLE FRAME PROCESSING AND CLASSIFICATION ENGINE

BACKGROUND OF THE INVENTION

FIELD OF INVENTION

[0001] The present invention relates to the handling of data arriving at a communications device, such as a network switch that directs the flow of that data. More specifically, the method and apparatus allow for the application of rules to the data to be handled such that the data can be classified, filtered and processed.

DESCRIPTION OF RELATED ART

[0002] A high speed network utilizes network devices to control data flowing through a network, where the network devices receive incoming data and pass the data on to other network devices. The passage of the data through the network is accomplished by examination of at least a portion of the incoming data. These network devices are often required to perform many functions, including packet or frame classification, packet processing or filtering, quality of service (QoS) or class of service (CoS) enforcement, application server load balancing, and statistics data gathering for packet traffic and processing.

[0003] Classification of frames or packets is an important process in many network device systems such as routers, switches, bridges, hubs, and aggregators. Packets may be identified as being in certain classes based on their source, destination, bandwidth requirements, and application types. Different classes of packets can receive differentiated processes such as different latencies, transmission rate, and routing paths to provide most desirable services to all packets. Filtering of packets is a crucial process of ensuring network security such as with a firewall or a way of forwarding packets only to intended destinations. Packet filtering performs the granting of access or the blocking of packets for certain application types and protocols. It can also allow or block packets' access to a network based on their subnet, group of hosts, or individual host information. The information of QoS (Quality of Service) or CoS (Class of Service) is usually embedded in the header of packets such as VLANID (Virtual Bridged Local Area Network Identification), TOS (type of service), DiffServ (Differentiated Service) field information.

[0004] The prior art network devices can provide such classification of frames or packets, but those classification processes employed by the devices are often cumbersome and difficult to make changes thereto. The prior art devices are also limited in how they can be customized and provide limited options for how data should be handled once a classification match is made. Also, because of the processes applied by the prior art network devices in classification, the throughput of a network can be negatively affected.

[0005] Thus, there is a need for a processing and classification engine that allows the engine to be customized without underutilizing the processing abilities of the switch. Additionally, there is also a need for a switch architecture that allows for expandability and continued customization after the sale of the initial switch configuration.

SUMMARY OF THE INVENTION

[0006] It is an object of this invention to overcome the drawbacks of the above-described conventional network devices and methods. The present invention provides for efficient classification, filtering, and processing of data frames or packets in networking systems. The present invention allows for a large number of complex rules to be expressed in a compact but highly flexible form. Thus, the present invention provides for low cost and high performance silicon solutions of efficient rule-based frame processing and classification, often called policy engines or classification engines.

[0007] The present invention is directed to a method of applying a set of rules to incoming data and apparatus that implements the efficient method. All the rules in the rule table are compared with the target packet either in parallel or one rule at a time. Once all these comparisons are finished, it is determined which rule matches the packet. If more than one rule match the packet, the rule of the highest priority is chosen as the final match.

[0008] According to one aspect of this invention, a method of handling data packets in a network device is disclosed. The method includes receiving an incoming data packet and the incoming data packet is parsed to obtain a portion of the incoming data packet. That portion is compared with rules stored in a rule table, where each rule specifies a set of actions. A match between the portion and a particular rule of the rules is selected and a particular set of actions, specified by that particular rule is executed. Each rule includes a mask and a selection flag that are used in the comparison of the portion with each rule.

[0009] Additionally, the rules may be compared with the packet portion serially or in a parallel fashion. If more than one rule matches the portion, the highest priority is selected as the matching rule, where this highest priority rule can be determined by the addresses of the rules within the rules table. The rules table can be implemented in a static random access memory or in a content addressed memory.

[0010] When comparing the portion with rules stored in a rule table, the process may also include applying the mask of one of the rules to the portion to obtain a packet field value. The packet field value is then compared with a rule field value contained in the one of the rules and the selection flag of the one of the rules is examined to determine whether the one of the rules is a potential match. Additionally, the examination of the selection flag can include inverting the result of the packet field value comparison when the selection flag is set to a particular value. Also, the validity of packet field value can be determined and applied in the determination whether that one rule is the potential match.

[0011] When the particular set of actions specified by the particular rule are executed, these actions can include modifying a header of the incoming data packet, forwarding the incoming data packet to a destination address, or updating a management information register. This can also include setting a flow identification for the incoming data packet such that the packet is classified according to a class of service.

[0012] In another aspect of the invention, the above methods are performed by an apparatus used to handle data packets in a network device. The apparatus includes means for receiving an incoming data packet and the incoming data packet is parsed by a means for parsing to obtain a portion of the incoming data packet. That portion is compared with rules stored in a rule table, where each rule specifies a set of actions. A match between the portion and a particular rule is selected by a means for selecting and a particular set of actions, specified by that particular rule is executed by an executing means. Each rule includes a mask and a selection flag that are used in the comparison of the portion with each rule.

[0013] These and other objects of the present invention will be described in or be apparent from the following description of the preferred embodiments.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] For the present invention to be easily understood and readily practiced, preferred embodiments will now be described, for purposes of illustration and not limitation, in conjunction with the following figures:

[0015] Fig. 1 shows an exemplary rule table of size *N* and the structure of each rule and actions;

[0016] Fig. 2 describes different types of rules with examples thereof;

[0017] Fig. 3 shows comparison processes for the 3 rule types. Fig. 3(a) illustrates an example of a IPDA field for rule type 1, Fig. 3(b) illustrates an example of a TDA field for rule type 2, and Fig. 3(c) illustrates an example of a PRO A field for rule type 3;

[0018] Fig. 4 illustrates one validity checking method, which can save extra bit fields for valid bits at the cost of all-zero detection logic;

[0019] Fig. 5 illustrates another validity checking method, which needs extra bit fields for valid bits but eliminates all-zero detection logic;

[0020] Fig. 6 describes 3 types of actions in a rule and their examples;

[0021] Fig. 7 shows an overall structure of an SRAM-based implementation of one embodiment of the present invention, where an SRAM stores rules and actions, and a number of comparison logic units are running simultaneously on multiple packets; and

[0022] Fig. 8 shows an overall structure of a CAM-based implementation of one embodiment of the present invention, where a CAM stores rules and compares them with a packet header, and actions are stored in a separate SRAM addressed by the CAM's matched index.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[0023] The present invention provides a method of classification, which is highly flexible so it can be applied to any arbitrary classification engine. Packets may be identified as being in certain classes based on their source, destination, bandwidth requirements, and application types. Different classes of packets can receive differentiated processes such as different latencies, transmission rate, and routing paths to provide most desirable services to all packets. Filtering of packets is a crucial process of ensuring network security such as with a firewall or a way of forwarding packets only to intended destinations. Packet filtering performs the granting of access or the blocking of packets for certain application types and protocols. It can also allow or block packets' access to a network based on their subnet, group of hosts, or individual host information. One of applications of the present invention is to modify the QoS or CoS information by overriding VID or TOS/DiffServ field values or rewriting the CoS values of a packet to meet the enterprise policy. The present invention can provide an efficient method of collecting statistics of packet processing by triggering a set of events to management information base counters such as RMON II (remote network monitoring) and SLA (service level agreement) counters. Thus, the present invention also allows gathering of protocol-host based statistics and provides accounting statistics per arbitrary groups per CoS uses.

[0024] In one embodiment of the present invention, a frame processing and classification engine, is realized as a rule table that consists of a number of rules and actions. The present invention provides for a novel way of organizing the rules and actions in a regular and highly flexible fashion, and a mechanism by which the rule comparison and its validity checking are conducted.

[0025] All the rules in the rule table are compared with the target packet either in parallel or one rule at a time. Once all these comparisons are finished, it is determined which rule matches the packet. If more than one rule match the packet, the rule of the highest priority is chosen as the final match. One way of assigning priorities to rules is to treat rules at lower addresses as higher priorities. This way, expensive comparators for priority values can be avoided in the classification engine design.

[0026] Each rule specifies a set of field values to compare. In addition, each field in a rule involves a mask and equality/inequality selection flag called an equal bit. Each rule also specifies a set of actions, which are executed when the rule matches the target packet. Rules can be stored in an SRAM or a specially designed content-addressed memory (CAM). Fig. 1 shows an example of rule table 100 having *N* rules. It also illustrates example rule fields 101-103 and action fields 111-113.

[0027] The fields in a rule table can be either fixed to certain fields in the packet header such as IPDA or TDS, or programmed to be any contiguous bytes starting from a certain address in the packet header. For example of the latter case, the

VLANID can be chosen by setting the start address of a programmable field to 15th Byte in the packet header.

[0028] The mask of each field allows the comparison of the rule field value and the packet's field value to be limited to specified bits. For example, when an 8-bit mask is set to 11111000 in binary, only the first five bits of the field values are compared, and the other three bits are considered as identical or ignored.

[0029] The equal bit inverts the result of comparison, when set to zero. That is, a rule can be a match, when the comparison of a certain field turns out to be not equal but its equal bit is zero. This equal bit is very useful for comparing a range of values or identifying packets whose source or destination address does not meet certain conditions.

[0030] A rule is determined to be a match, when all its fields with the equal bit set to 1 match the corresponding fields in the packet header after applying their masks, and all its fields with equal bit set to 0 do not match the corresponding fields in the packet header after applying their masks. After comparing all of the rules with the packet header, only one rule is determined to be the final match. If there are multiple rules matched, the rule with the lowest address is selected as the final match as described above.

[0031] When a matched rule is determined, all of the actions specified in the rule are executed to either direct the packet to a specific destination or modify the packet's header fields. Actions can also specify how and which management information can be updated and indicate the disposition of the packet as a result of the match. In addition, a flow ID of the packet can be specified as an action field, which allows the present invention to be used as a frame classification engine as well as a frame processing engine.

[0032] In this example, the rule fields are categorized into three types. Rule fields in type 1 have a fixed location and a compact mask for each field, so they can efficiently handle long fields in a packet's header such as IPDA and IPSA, which are illustrated in Fig. 2(a). For example, an IPDA or IPSA field value is assigned to a 32bit bit field in each rule, but the 32bit mask is encoded to a 5bit number. An encoding 0 corresponds to a 32bit mask of all 1s, an encoding 1 to a 32bit mask of all 1s except the LSB (least significant bit), and an encoding 2 to a 32bit mask of all 1s except the last 2 bits, and so forth. This encoding is effective and sufficient for most IPDA or IPSA cases, because it is very common that only the MSB (most significant bit) portion of IP addresses are compared to identify a packet's destination or source.

[0033] A prevalent IP address search method is called a longest prefix match (LPM) and is also based on this fact. In order to represent a 32bit mask with all 0s, an additional bit field called Mask_All is allocated in the rule. The equal bit in the rule reverses the result of comparison when set to zero, while it passes the match result as it is when set to 1, as described above. Fig 3(a) illustrates the process of comparison for the type 1 rule fields.

[0034] In Fig. 3(a), the small-bit mask and the Mask_All bit field are encoded to form the large-bit mask in 301. The comparison value is bit-wise ANDed with the large-bit mask in 302 and a similar ANDing is performed on the large-bit mask and the IPDA or IPSA of the packet in 303. Both results are compared in 304 and the result is XORed 305 with the equal bit to determine if a match exists.

[0035] Rule fields in type 2 have a fixed field location, but have a full mask which is as wide as the field value; it is used for short fields such as TDS, TSS, PROT, TOS, TYPE, VLANID, and VPRIO (VLAN PRIORITY). An example of these type 2 rule fields are illustrated in Fig. 2(b). Most of these fields in a packet header do not have the property of IP address's prefix match, and so need full mask. The equal bit works in the same way as in the rule field type 1 case. Fig. 3(b) illustrates an example of the process of comparison for the type 2 rule fields.

[0036] In Fig. 3(b), the full mask is input into 312 and 313. The comparison value is bit-wise ANDed with the full mask in 312 and a similar ANDing is performed on the full mask and the TDS, etc., of the packet in 313. Both results are compared in 314 and the result is XORed 315 with the equal bit to determine if a match exists.

[0037] Rule fields in type 3 have a programmable field location, so they can be mapped to any contiguous 8 or 32bit values in a packet header. For example, each rule has four 8bit programmable fields, PRO A, PRO B, PRO C, and PRO D, and two 32bit programmable fields, PRO E and PRO F. Each rule field of this type has a global programmable start byte address as shown in Fig. 2(c).

[0038] For example, StartByte A points to the start byte address for all PRO A field in every rule in the rule table. All the programmable rule fields have an equal bit which works in the same way as in rule field type 1 and 2. In addition, there is a global programmable flag that indicates whether the byte address 0 is the first byte of the Layer 2 fields or the first byte of the Layer 3 fields in the packet header. Depending on how this flag is set, the six programmable rule fields can pick up different values either starting from a Layer 2 field (MACDA field) or starting from a Layer 3 field (VERSION field) in the packet header. The range of addresses that a programmable rule field can cover depends on how many bytes a header parser reads from the packet header for parsing. For example, it can be implemented to cover up to 64 Bytes starting from the first byte of Layer 2 address. Figure 3(c) illustrates the process of comparison for the type 3 rule fields.

[0039] In Fig. 3(c), the start byte address is given along with the entire header of a packet to 321 to provide a specified field of the packet. The full mask is input into 322 and 323 and the comparison value is bit-wise ANDed with the full mask in 322 and a similar ANDing is performed on the full mask and the specified field in 323. Both results are compared in 324 and the result is XORed 325 with the equal bit to determine if a match exists.

[0040] Validity of each header field can be determined according to the following two methods. In the first method, a packet header parser determines whether or not each field in a packet header is valid, and set its valid bit to 1 or 0, respectively. These valid bits are sent by the header parser to the rule table along with the packet field values. For example, if a TCP packet is fragmented into a few IP packets, the layer 4 fields, such as TSS and TDS, in the fragmented IP packet, except in the first packet, are all invalid, and their valid bits are set to 0. Also if the packet header parser recognizes only IPv4 header format but a target packet contains a layer 3 format other than IPv4 (internet protocol version 4), the layer 3 fields of the header , such as IPDA and IPSA, are all set to invalid.

[0041] Another validity decision, which is particularly unique to the present invention, is how to determine the address validity of programmable fields when they are beyond the range the parser can handle. Suppose the header parser reads the packet header only up to the first 64 bytes. If an 8bit programmable field such as

PRO A in Fig. 2(c) is pointing to an address beyond the first 64 Bytes, the PRO A field is automatically invalidated, and its valid bit is set to 0. If a 32bit programmable field such as PRO E in the above example is beyond the first 64 Bytes, a decision is made as follows. A 3bit valid flag is used to indicate how many bytes (0 to 4 Bytes) in the 32bit programmable field are valid. This flag tells how many bytes in the 32bit field starting from its MSB (most significant bit) are within the first 64 Bytes and so are valid. For example, if 3 bytes of the 32bit field are within the first 64 Bytes, the valid flag is set to 3.

[0042] The second method of determining a field's validity is to use programmable fields and compare them with certain packet field. For example, the rule table can use PRO A field to compare the packet's VERSION field. If VERSION equals 4, the layer 3 fields conform to the IP format, so all IP fields such as IPDA, IPSA, PROT, and TOS are considered as valid. Otherwise, all IP fields are invalid; that is, a rule is a match only when the rule's PRO A field is a match, meaning that the rule is valid. For another example, PRO E field can be programmed to compare it with the EtherLength field. If EtherLength equals 8100 in hexadecimal, the VLAN fields are valid. Otherwise, the VLAN fields are invalid. Thus, by setting the PRO E field to 8100, the validity of certain rules can be set.

[0043] The first method of determining field validity leads to an efficient usage of rule fields, because all the validity decisions are made by the header parser, and the parser provides the rule table with a valid bit or flag for every field. Thus no programmable fields are dedicated to validity determination. The second method of determining validity requires a simpler comparison logic in the frame classification and processing engine because rules that are determined as invalid by dedicated programmable fields always lead to no match, so it does not need the extra valid bits and flags to determine which rules are matched. However, this benefit comes at the cost of dedicating some of the programmable fields to validity decision, sacrificing its flexibility.

[0044] When the first validity decision method is employed, two methods of validity checking for each field may be used. Fig. 4 shows how the validity checking method 1 works. It is noted that elements 402-405 are equivalent to elements 302-305, 312-315 or 322-325 of Fig. 3, depending on the type of rule field that is being validated. Validity checking method 1 uses the mask of each field to check whether to invalidate the match result of the field. When the packet header field is invalid (valid bit from the header parser is 0), unless the mask is all 0s 406 for rule field type 2 and 3, or the Mask_All bit is 1 for rule field type 1, the match result is overridden by the result of 407 and set to 0, regardless of the value of equal bit, in 408. When the valid bit is 1, the match result and the equal bit contents are always honored. This way, the frame classification and processing engine can limit its search to rules whose packet header fields are valid or whose field masks are all 0s (masked out). The validity checking for each field is conducted independently, so a rule that contains an invalid field still can be a match, if its other fields are valid and match the corresponding packet header fields.

[0045] Fig. 5 shows the process of validity checking method 2, which is also aimed at the first validity decision method described above. It is noted that elements 502-505 are equivalent to elements 302-305, 312-315 or 322-325 of Fig. 3, depending on the type of rule that is being validated. Instead of using masks, it employs extra bit fields for valid bits for each rule field in a rule. If the valid bits for certain rule fields in a rule is set to 0, the engine considers these rule fields to be matched, 508, only when the corresponding valid bits from the header parser are 0 (invalid), and all valid rule fields in the rule match their packet header fields, 506. The method 2 allows a rule to be used for packets with a certain field valid or packets with the fields invalid, but not both. Thus, it provides a higher flexibility than method 1. On the other hand, the method 1 tends to require fewer rules, because it allows one rule to be used for both packets with a certain field valid and packets with that field invalid.

[0046] The method 1 is well suited to a frame classification and processing engine implementation using an SRAM, because the comparison logic is outside the memory cell array and so can easily incorporate the extra all-zero detection logic for validity checking. On the other hand, the method 2 can be employed for a frame classification and processing engine implementation using either CAM or SRAM, because it does not need the all-zero detection logic; integrating all-zero detection logic in every entry of common CAM structures is very expensive.

[0047] As discussed above, there are three types of actions for each rule. Fig. 6 shows the 3 types of actions and examples thereof. Actions in type 1 are packet disposition commands represented by 1 bit field for each command. For example, the Drop action field, when set to 1, indicates to drop the packet when the packet buffer is congested or almost full, while the Copy action field, when set to 1, indicates to copy the packet to a special management unit such as CPU interface for further process of the packet. The Forward action field tells to forward the packet even if the buffer is congested. Actions in type 2 are commands to replace certain field values. These actions consist of a 1bit flag indicating whether to replace or not, and a new field value. These actions can specify new field values such as destination channel or port, DSCP (differential service code point), VLAN ID, and VLAN PRIO (priority). Actions in type 3 are special commands that provide a bit map such as management information statistics control signals or a certain ID for classification of the packet under process. For example, a type-3 action can provide an 8bit bitmap that indicates to increment 8 individual counters in a management information base. Also a type-3 action can specify a flow ID of the packet, which can be used to classify and process the packets depending on their priorities and an endowed bandwidth limit.

[0048] The rule comparison mechanism of the present invention can be implemented either using an SRAM or a CAM. A design with an SRAM is illustrated in Fig. 7. This design uses a number of separate comparison logic blocks 703-705 that are running simultaneously in order to reduce the comparison or search latency. For example, when a rule table 701 with 64 rules is implemented using a SRAM with an address range of 0 to 63 and only one comparison logic, the latency of a complete search is 64 times the SRAM's read cycle time. If 4 sets of comparison logic are employed, the latency can be reduced by a factor of 4.

[0049] An implementation with a CAM is given in Fig. 8. This implementation uses a special CAM 801 architecture to allow the comparison logic of CAM to determined a match result based on valid and equal bits in each rule. This involves an extra bit fields (valid bit and equal bit) in every CAM entry and an extra XOR logic in addition to the comparison logic of a common CAM architecture.

[0050] Since it is expensive to integrate an all-zero detection logic in every entry of a CAM as described above, the validity check method 2, illustrated in Fig. 5, for the CAM-based implementation can be used. In contrast, the validity checking method 1 in Fig. 4 is well suited to the SRAM-based implementation, because it requires only a few all-zero detection logic in each comparison logic.

[0051] The above-discussed configuration of the invention is, in one embodiment, embodied on a semiconductor substrate, such as silicon, with appropriate semiconductor manufacturing techniques and based upon a circuit layout which would, based upon the embodiments discussed above, be apparent to those skilled in the art. A person of skill in the art with respect to semiconductor design and manufacturing would be able to implement the various modules, interfaces, and tables, buffers, etc. of the present invention onto a single semiconductor substrate, based upon the architectural description discussed above. It would also be within the scope of the invention to implement the disclosed elements of the invention in discrete electronic components, thereby taking advantage of the functional aspects of the invention without maximizing the advantages through the use of a single semiconductor substrate.

[0052] It is noted that while the present invention cites, as one example, a policy engine, the present invention is not limited to policy-based frame processing and classification. Although the invention has been described based upon these preferred embodiments, it would be apparent to those of skilled in the art that certain modifications, variations, and alternative constructions would be apparent, while remaining within the spirit and scope of the invention. In order to determine the metes and bounds of the invention, therefore, reference should be made to the appended claims.